Interfacing the Storage Resource Broker (SRB) to the Hierarchical Resource Manager (HRM)

Arie Shoshani, Alex Sim (LBNL)
Reagan Moore, Bing Zhu (SDSC)

PPDG meeting
July 13-14, 2000

Outline

- HRM role in the Data Grid architecture
- HRM API description
- HRM - HPSS system functionality
- SRB Client calls
- SRB-HRM architecture
- Status

HRM role in the Data Grid architecture

- The class of Storage Resource Managers (SRMs) includes:
  - HRM: for managing the access to tape resources
    - may or may not have a disk cache
    - functionality generic but needs to be specialize for specific mass storage systems
    - e.g. HRM-HPSS, HRM-Enstore, ...
  - DRM: for managing disk resources
    - functionality generic but needs to be specialize for specific disk systems
    - e.g. DRM-FileSystem, DRM-DPSS, …

HRM role in the Data Grid architecture

Functionality Examples

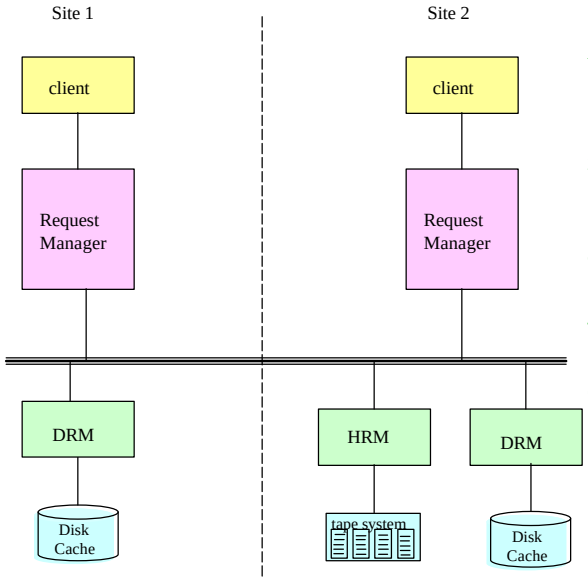
HRM functionality may include :

-- queuing of file transfer requests (kind of a reservation)
-- reordering of requests to optimize PFTP order (ordered by files on the same tape)
-- Monitoring progress and error messages
-- rescheduling transfers that failed

DRM functionality may include:

-- keeping tracks of files in cache
-- managing space reservations
-- making decisions on which files to remove when space is needed
-- optimizing cache use - sharing files requested by multiple clients
-- enforce local policy for cache use

HRMs & DRMs in a Grid Architecture



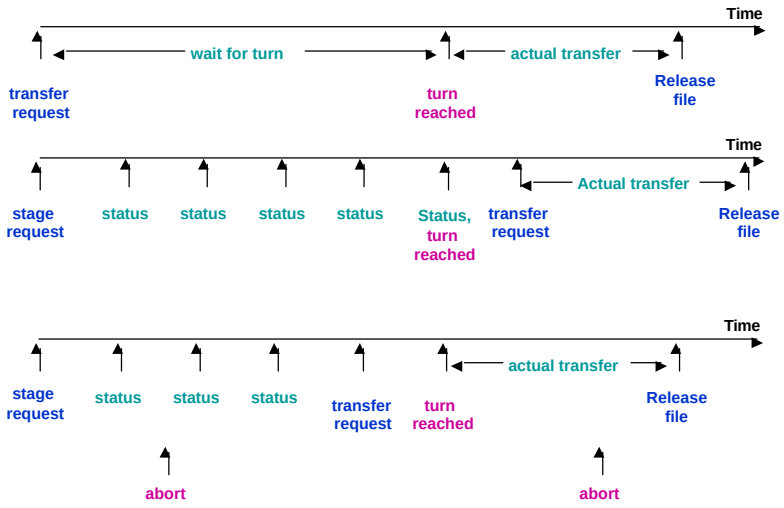
Advantages:

1. Each SRM has global view of all requests to it.
2. SRMs support reservation requests for space and file transfers
3. Can develop uniform error handling codes for all resources
4. Can ask SRMs to queue and monitor requests.
5. Main implication: need APIs to invoke SRMs, and a way for the Grid to pass requests to them.

HRM API description

- **API Functionality**
  - **Request to transfer a file to destination disk**
    - **A blocking call**
  - **Request to stage a file to HRM disk**
    - **A non-blocking call when HRM disk exists**
  - **Request status/time_estimate**
    - **How long before file request will be processed**
  - **Request to abort a file transfer or stage**
    - **In case file no longer needed**
  - **Release a file**
    - **After file was moved to destination**
    - **optional to improve system efficiency**
  - **Call back when file is staged**

Examples of call sequences



CORBA IDL for HRM

```
// hrm.idl
// IDL for HPSS Resource Manager
//
#ifndef HRM_IDL
#define HRM_IDL

enum hrmStatus {
        HRM_TRANSFER_COMPLETED,
        HRM_TRANSFER_FAILED,
        HRM_STAGE_ACCEPTED,
        HRM_STAGE_COMPLETED,
        HRM_STAGE_REFUSED,
        HRM_STAGE_FAILED,
        HRM_RELEASE_DONE,
        HRM_RELEASE_FAILED,
        HRM_ABORT_DONE,
        HRM_ABORT_FAILED,
        HRM_FILE_DOES_NOT_EXIST,
        HRM_HPSS_DOWN,
        HRM_HPSS_ERROR
};

module HPSSResourceManager {
        interface hrmGrid {
                hrmStatus transferFile
                        (in string sourceUrl,
                         in string destinationUrl,
                         in long waitTime,
                         in short userPriority,
                         out double timeEstimate);
                hrmStatus stageFile
                        (in string reference,
                         in string destinationUrl,
                         in long waitTime,
                         in short userPriority,
                         out double timeEstimate);
                boolean abort(in string destinationUrl);
                double timeEstimate(in string destinationUrl);
                boolean releaseFile(in string destinationUrl);
        };
};
#endif
```

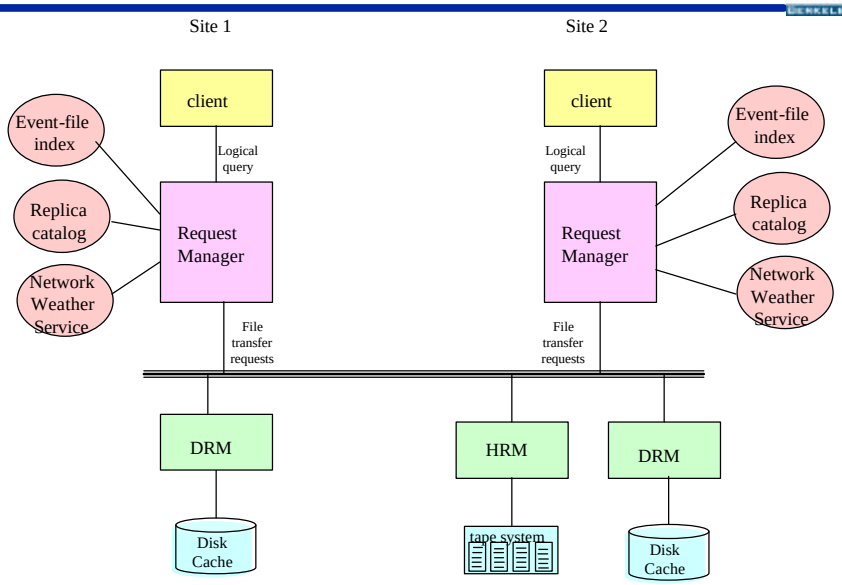
HRM-HPSS system functionality

- All transfers go through HRM disk
  - reasons: flexibility of pre-staging
  - disk is sufficiently cheap for a large cache
  - opportunity to optimize for same file requests
- Functionality
  - queuing file transfers
  - file queue management
  - File clustering parameter
  - Transfer rate estimation
  - Query estimation - total time
  - Error handling

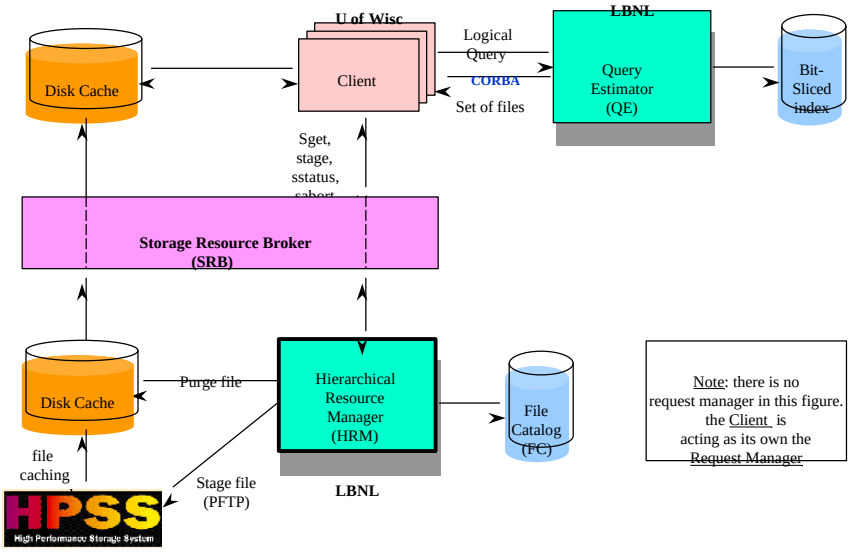
SRB Client calls

- Sget
  - blocking call (wait till file is transferred)
  - should only be issued if space on client disk is allocated
- Sstage
  - non-blocking call (returns time estimate)
  - requires a server if want to be notified
  - can issue status to find out if file was cached
- Sstatus
  - returns time estimate for file to be staged to HRM's disk
- Sabort
  - cancel this file request

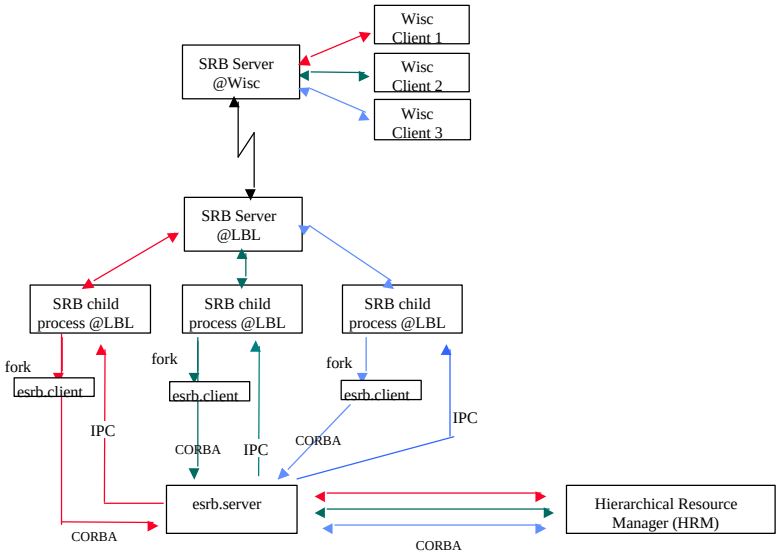
PPDG Grid Architecture

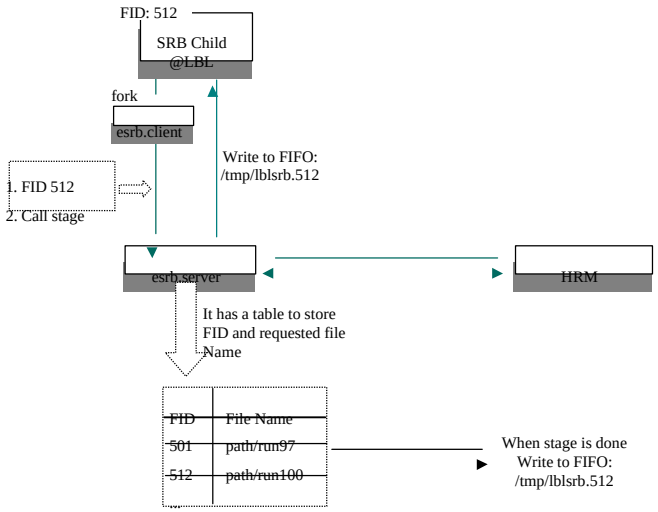


Current PPDG test setup



SRB-HRM interaction





Status

- Functionality that was tested
  - request a file transfer - Sget
  - A single Client
- Functionality that exists but was not tested
  - Sget, Sstage, Sstatus
  - multiple Clients concurrent requests
  - use of replica catalog developed by SDSC
- Functionality that does not exist
  - A request manager that uses the replica catalog
  - A request manager that accesses multiple sites
  - A request manager that makes informed choices
  - A disk resource manager

Optional Slides

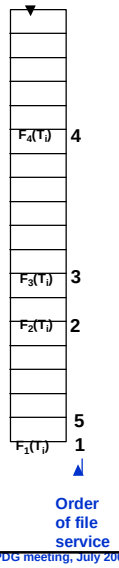
- The following slides shows how HRM implements the functionality it provides
- They will be shown only if time permits

Queuing File Transfers

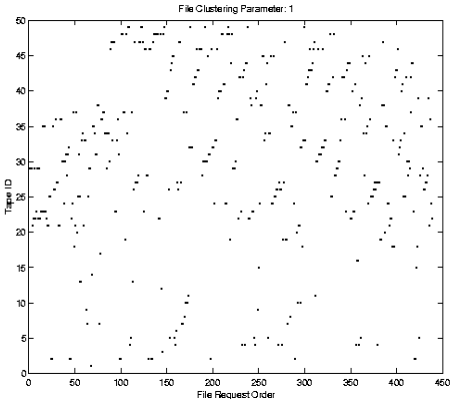
- Number of PFTPs to HPSS are limited
  - limit set by a parameter - No_PFTP
  - parameter can be changed dynamically
- HRM is multi-threaded
  - issues and monitors multiple PFTPs in parallel
- All requests beyond PFTP limit are queued
- File Catalog used to provide for each file
  - HPSS path/file_name
  - Disk cache path/file_name
  - File size
  - tape ID

File Queue Management

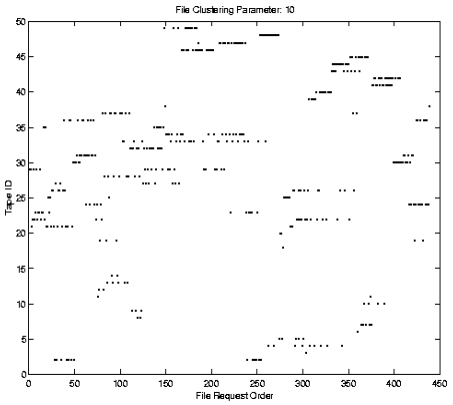
- Goal
  - minimize tape mounts
  - still respect the order of requests
  - do not postpone unpopular tapes forever
- File clustering parameter - FCP
  - If the file at top of queue is in $Tape_i$ and FCP > 1 (e.g. 4) then up to 4 files from $Tape_i$ will be selected to be transferred next
  - then, go back to file at top of queue
- Parameter can be set dynamically



File Caching Order for different File Clustering Parameters



File Clustering Parameter = 1



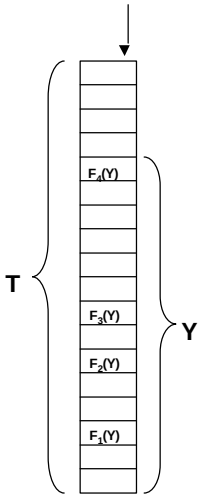
File Clustering Parameter = 10

Transfer Rate (Tr) Estimates

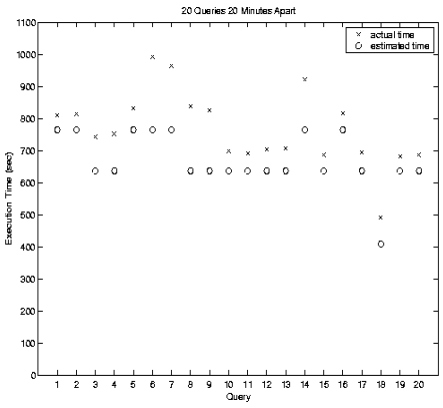
- Need Tr to estimate total time of a query
- Tr is average over recent file transfers from the time PFTP request is made to the time transfer completes. This includes:
  - mount time, seek time, read to HPSS Raid, transfer to local cache over network
- For dynamic network speed estimate
  - check total bytes for all file being transferred over small intervals (e.g. 15 sec)
  - calculate moving average over n intervals (e.g. 10 intervals)

Query Estimate

- Given: transfer rate Tr.
- Given a query for which:
  - X files are in cache
  - Y files are in the queue
  - Z files are not scheduled yet
- Let s(file_set) be the total byte size of all files in file_set
- If Z = 0, then
  - QuEst = s(Y)/Tr
- If Z ≠ 0, then
  - QuEst = (s(T)+ q.s(Z))/Tr
    where q is the number of active queries

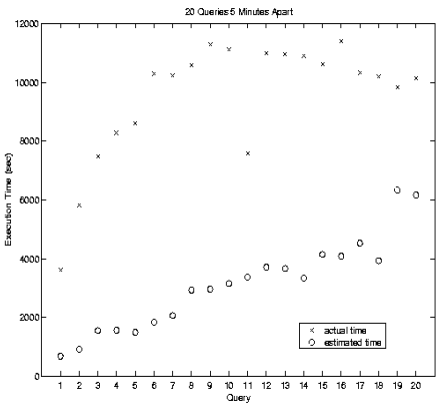


Reason for q.s(Z)



20 Queries of length ~20 minutes launched 20 minutes apart

Estimate pretty close



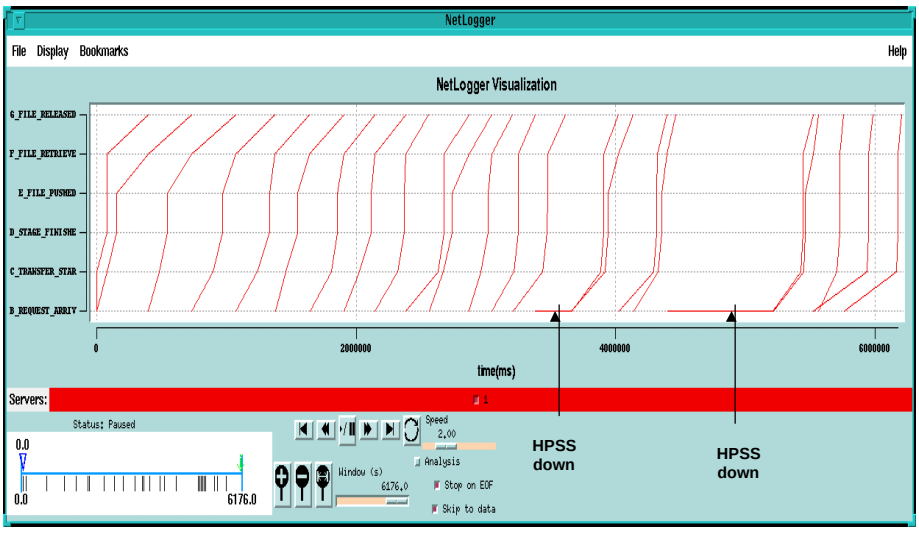
20 Queries of length ~20 minutes launched 5 minutes apart

Estimate bad - request accumulate in queue

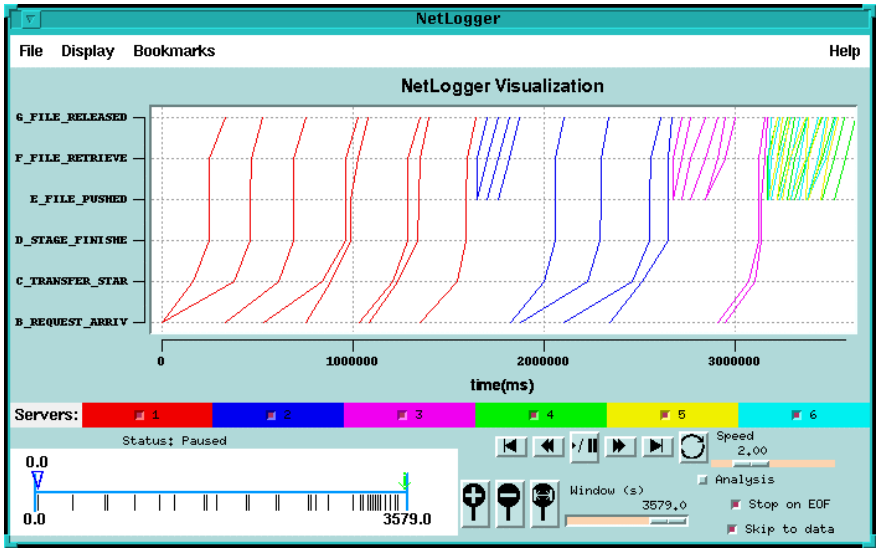
Error Handling

- 5 generic errors
  - — file not found
    - return error to caller
  - — limit PFTP reached
    - can't login
    - re-queue request, try later (1-2 min)
  - — HPSS error (I/O, device busy)
    - remove part of file from cache, re-queue
    - try n times (e.g. 3), then return error "transfer_failed"
  - — HPSS down
    - re-queue request, try repeatedly till successful
    - respond to File_status request with "HPSS_down"

File Tracking (1)



File Tracking (2)



Dynamic Display of Various Measurements